Detecting Hunts in Wildlife Documentaries

Niels C. Haering, Richard J. Qian, M. Ibrahim Sezan, and Niels da Vitoria Lobo

N. C. Haering and N. Lobo are with the University of Central Florida, Orlando, Florida, U.S.A.
R. J. Qian and M. I. Sezan are with Sharp Labs of America, Camas, Washington, U.S.A.

Abstract

We propose a multi-level video event detection methodology and apply it to animal hunt detection in wildlife documentaries. The proposed multi-level approach has three levels. The first level extracts color, texture, and motion features, and detects moving object blobs. The mid-level employs a neural network to verify whether the moving object blobs belong to animals. This level also generates shot descriptors that combine features from the first level and contain results of mid-level, domain specific inferences made on the basis of shot features. The shot descriptors are then used by the domain-specific inference process at the third level to detect the video segments that contain hunts. The proposed approach can be applied to different domains by adapting the mid and high-level inference processes. Event based video indexing, summarization and browsing are among the applications of the proposed approach.

Keywords

Video content analysis; content-based indexing and retrieval; browsing and visualization.

I. Introduction

The amount of video information that can be accessed and consumed from people's living rooms has been ever increasing. This trend may be further accelerated due to the convergence of both technology and functionalities supported by future television receivers and personal computers. To obtain the information that is of interest and to provide better entertainment, tools are needed to help users to extract relevant content and to effectively navigate through the large amount of available video information. For ordinary users, such tools may also have to satisfy the following requirements: (1) they should be easy to use in terms of operations; and (2) they should be easy to understand and predict in terms of behaviors.

Existing content-based video indexing and retrieval methods do not seem to provide the tools which are called for in the above applications. Most of those methods may be classified into the following three categories: (1) syntactic structurization of video; (2) video classification; and (3) extraction of semantics. The work in the first category has concentrated on (a) shot boundary detection and key frame extraction, e.g., [1], [33]; (b) shot clustering, e.g., [31]; (c) table of content creation, e.g., [9]; (d) video summarization, e.g., [20]; and (e) video skimming [26]. These methods are in general computationally simple and their performance is relatively robust. Their results, however, may not necessarily be semantically meaningful or relevant since they do not attempt to model and estimate the

semantic content of the video. For consumer oriented applications, semantically irrelevant results may distract the user and lead to frustrating search or browsing experience. The work in the second category tries to classify video sequences into certain categories such as news, sports, action movies, close-ups, crowd, etc. [17], [28]. These methods provide classification results which may facilitate users to browse video sequences at a coarse level. Video content analysis at a finer level is probably needed, to more effectively help users find what they are looking for. In fact, consumers often express their search items in terms of more exact semantic labels, such as keywords describing objects, actions, and events. The work in the third category has been mostly specific to particular domains. For example, methods have been proposed to detect certain events in (a) football games [16]; (b) soccer games [32]; (c) basketball games [25]; (d) baseball games [18]; and (e) sites under surveillance [5]. The advantages of these methods include that the detected events are semantically meaningful and usually significant to users. The major disadvantage, however, is that many of these methods are heavily dependent on specific artifacts such as editing patterns in the broadcast programs, which makes them difficult to extend for the detection of other events. A query-by-sketch method has also been proposed recently in [2] to detect certain motion events. The advantage of this method is that it is domain-independent and therefore may be useful for different applications. For consumer applications, however, sketching needs cumbersome input devices, specifying a query sketch may take undue amounts of time and learning the sketch conventions may discourage users from using such tools.

In this paper, we propose a computational framework and several algorithmic components towards an extensible solution to semantic event detection. The automated event detection algorithm may enable users to effectively find certain semantically significant events in their video content and help generate semantically meaningful highlights for fast browsing. In contrast to most existing event detection work, our goal is to develop an extensible computational approach which may be adapted to detect different events in different domains. To achieve this goal, we propose a three-level video event detection algorithm. The first level extracts color, texture, and motion features, and detects moving object blobs. The mid-level employs a neural network to verify whether the moving blobs

belong to objects of interest. This level also generates shot descriptors that combine features from the first level and contain results of mid-level, domain specific inferences made on the basis of shot features. The shot descriptors are then used by a domain-specific inference process at the third level to detect the video segments that contain events of interest. To test the effectiveness of our algorithm, we have applied it to detect animal hunt events in wildlife documentaries. In our implementation we do not attempt to detect the stalking phase that precedes many hunts. Our purpose is to detect the swift or rapid chase of a fleeing or running animal. Since hunts are among the most interesting events in a wildlife program, the detected hunt segments can be composed into a program highlight sequence. The proposed approach can be applied to different domains by adapting the mid and high-level inference processes while directly utilizing the results from the low-level feature extraction processes.

In the following section, we describe the proposed computational framework and its algorithmic components. In Section 3, we present experimental results obtained as we applied the proposed algorithm to detection of animal hunt events in a number of commercially available wildlife video tapes. Implementation details are also furnished in Section 3. Finally in Section 4, we summarize our work and discuss some future directions.

II. Methodology

We focus on the classification and detection of non-rigid, amorphous or articulate natural objects, such as animals, trees, grass, sky, clouds, etc., as well as the motion of objects in such scenes. Our approach therefore has object classification and motion detection components. The object classification component makes use of feature extraction methods based on multi-resolution Gabor filters, the Gray-Level Co-occurrence Matrix (GLCM), the fractal dimension, and color. The feature representations of the objects are then classified by a back-propagation neural network and combined with shot boundary information and frame motion estimates to detect semantic events such as predators hunting prey.

The problem of detecting semantic events in video, e.g., hunts in wildlife video, can be seen as having three levels as shown in Figure 1. At the lowest level we determine the boundaries between shots, estimate the global motion, and express each frame in a color and texture space. We also compensate for the estimated global motion between each pair

of frames. The earlier frame of each pair is transformed by the motion estimate and a difference image is produced to highlight areas of high residual error. We assume that this residual error is mostly due to independent object motion, and therefore the highlighted areas correspond to independently moving objects which are also referred as motion blobs.

At the intermediate level the detected motion blobs are then verified with the class labels assigned to that region by a neural network. The network uses the color and texture representation of the input obtained by the lower level, and performs a crude classification of image regions into sky, grass, tree, rock, and animal regions. If (1) the motion between two consecutive frames is large, (2) a blob exists that has a high motion residual (motion other than that of the background), and whose motion and position in consecutive frames varies smoothly, and (3) is labeled as animal region by the network then we assert that we are tracking a fast moving animal. The intermediate level generates and integrates such frame information and produces a summary for an entire shot. If throughout the shot there was support for a fast moving animal and the location/motion of the animal was found to be stable enough then the shot summary will indicate that a fast moving animal was tracked throughout the shot.

At the highest level a domain specific analysis of these shot summaries is used to infer the presence of a hunt in the underlying video sequence.

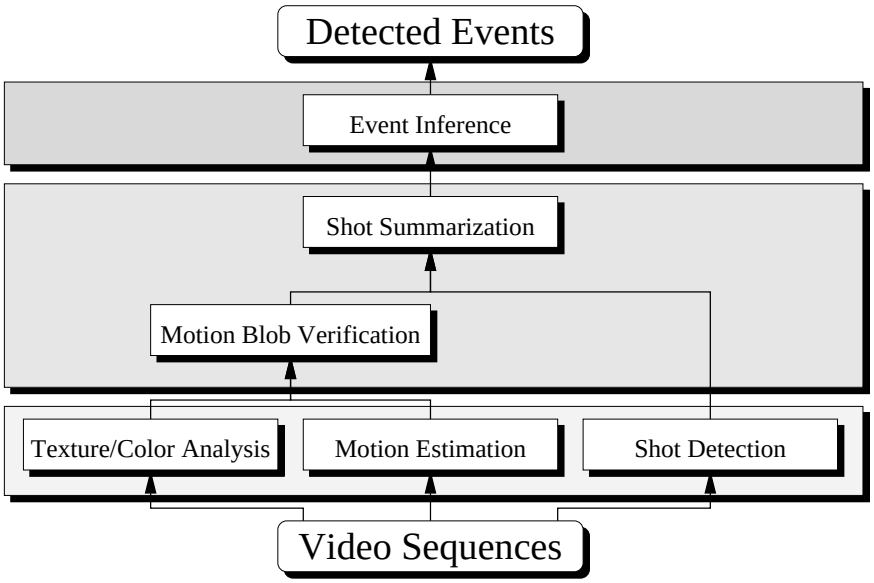


Fig. 1. The flowchart of our method.

A. Global Motion Estimation and Motion Blob Detection

We assume that the global motion can be estimated with a three parameter system allowing only for zoom, horizontal and vertical translation.

$$u(x, y) = a_0 + a_2 x$$
$$v(x, y) = a_1 + a_2 y$$

The robust recovery of the three parameters has to deal with the following problems,

- corresponding points in adjacent frames are often far apart (50-60 pixel displacements are not uncommon, peak displacements exceed 100 pixels),
- interlacing between frames drastically changes the appearance of small objects and textures in adjacent frames,
- the object and hence the global motion we are trying to estimate is often very large and motion blur eliminates texture in the direction of that motion (of course the motion in this direction is also the motion we are most interested in),
- often animals need to be tracked under strongly varying lighting conditions and occlusion, as when a hunt leads through areas with trees or bushes.

Given the large possible displacements between corresponding patches of adjacent frames an exhaustive search creates unreasonable processing requirements. Therefore we use a pyramid of reduced resolution representations of each frame. At each level of the 5-level pyramid we consider matches from a 5×5 neighborhood around the location of the patch in the source frame, enabling a maximum matching distance of 62 pixels. The levels of the pyramid are obtained by subsampling the lower level image rather than computing a more accurate Gaussian pyramid. We expect the use of a Gaussian pyramid to produce better results at a slight computational cost.

At the lowest level of the pyramid, i.e. the full resolution representation of the frame, the patches used for matching are of size 64×64. Patches from uniform areas often result in erroneous displacement estimates. To avoid matching such patches we discard patches with insufficient "texture". We use a 2D variance measure to determine the "amount of texture".

$$var_x = \sum_{j=0}^{n} (\sum_{i=0}^{m} (a(i, j) - a(., j))^2 - a(., .))^2$$

$$var_y = \sum_{j=0}^{n}(\sum_{i=0}^{m}(a(i,j) - a(i,.))^2 - a(.,.))^2$$

where $a(i,j)$ is an $m \times n$ image patch, $a(i,.), a(.,j)$, and $a(.,.)$ are the averages of the i^{th} row, the j^{th} column, and the entire patch a respectively.

We compute motion estimates at each of the four corners of a frame, as shown in Figure 5(a). Bad motion estimates are often due to matching errors made high up in the pyramid that are subsequently not recovered by the lower levels. Since the motion of the tracked animals often does not vary drastically between consecutive frames (i.e. their acceleration is small) we also use the previous best motion estimate to predict the location of the four patches in the next frame. A limited search in a 5×5 neighborhood around the predicted location, improves the motion estimates in many cases. Therefore we obtain up to eight motion estimates, one pyramid based estimate for each of the four patch locations, and one for each of the four estimates based on a limited search around the predicted match locations. Since some patches may not pass the "texture" test we may have fewer than eight motion estimates. The highest normalized dot product between a source patch $P1$ and matched patch $P2$ determines the "correct" global motion estimate between the current and next frame. The normalized dot product is equal to the cosine of the angle (α) between the two patches (vectors) $P1$, and $P2$:

$$cos(\alpha)_{P1,P2} = \frac{\sum_{i,j} P1(i,j)P2(i,j)}{\sum_{i,j} P1(i,j) \sum_{i,j} P2(i,j)}$$

We would like to point out that

- almost all wildlife videos are taken with a tele lens at a great distance to the objects of interest. For our motion analysis, we therefore assume an orthographic model, in which the camera pan and tilt appear as plain translations, thus supporting our assumption of uniform background motion,
- motion estimates based on the *feature space representation* of the frames are very similar to those obtained on the original *color* frames, and
- although the described motion estimation scheme is sufficient for our purpose a Kalman filter based approach [11] might yield more consistent results.

The motion estimates are then used to compensate for the global motion between consecutive frames. Finally, we use the grayvalue difference between the current image and

the motion compensated next frame to estimate the location of the animal in the frame. Areas with low residual error are assumed to have motion values similar to those of the background and are ignored. The independent motion of animals on the other hand usually causes high residual errors between the current frame and the following motion compensated frame. Therefore we can make use of a robust estimation technique to obtain an estimate of the animal location within the frame. This estimation technique iteratively refines the mean x and y values dependent on the residual error within a fixed size neighborhood around the mean values for the entire difference image. The robust estimation method was first developed in [24] for real-time human face tracking. Here we briefly describe how the method is applied to the application discussed in this paper. Based on the frame difference result, the algorithm constructs two 1D histograms by projecting the frame difference map along its x and y direction, respectively. The histograms, therefore, represent the spatial distributions of the motion pixels along the corresponding axes. Figure 2(a) illustrates an ideal frame difference map where there is only one textured elliptical moving object in the input sequence, and the corresponding projection histograms.

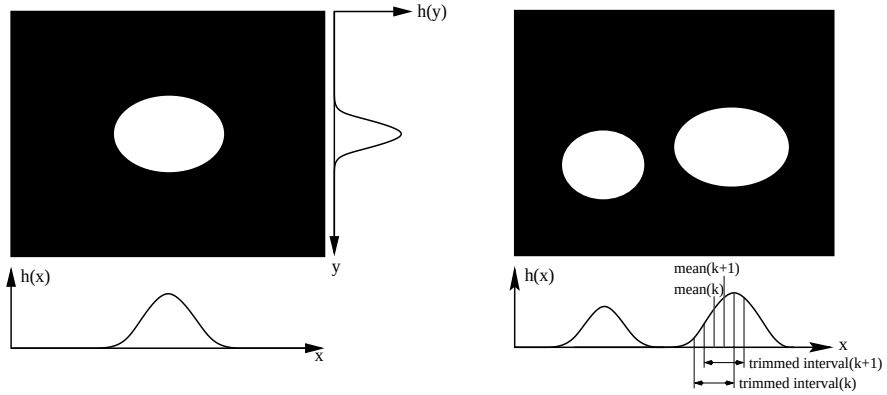


Fig. 2. (a) Two 1D histograms constructed by projecting the frame difference map along the x and y direction, respectively. (b) Robust mean estimation for locating the center position of a *dominant* moving object.

The instantaneous center position and size of a object in the image can be estimated based on statistical measurements derived from the two 1D projection histograms. For example, a simple method estimates the center position and size of a dominant moving object in an input sequence using the sample means and standard deviations of the dis-

tributions. More specifically, let $h_x(i), i = 0, 1, \ldots,$ and $h_y(i), i = 0, 1, \ldots,$ denote the elements in the projection histograms along the x and y direction, respectively. Then the object center position (x_c, y_c) and object width and height (w, h) may be estimated as:

$$x_c = \frac{\sum_i x_i h_x(i)}{\sum_i h_x(i)}, \quad y_c = \frac{\sum_i y_i h_y(i)}{\sum_i h_y(i)}, \quad w = \alpha \left[\frac{\sum_i (x_i - \mu_x)^2 h_x(i)}{\sum_i h_x(i)}\right]^{\frac{1}{2}}, \quad h = \beta \left[\frac{\sum_i (y_i - \mu_y)^2 h_y(i)}{\sum_i h_y(i)}\right]^{\frac{1}{2}}$$

where α and β are constant scaling factors.

However, the object center position and size derived from the sample means and standard deviations may be biased in the cases where other moving objects appear in the scene. It is therefore necessary to develop a more robust procedure to address this problem. We propose the use of robust statistical estimation routines to achieve robust measurements for object center position and size [30]. More specifically, the center position of a dominant moving object in an input sequence is estimated based on the robust (trimmed) means of the two 1D projection histograms in the x and y directions. Figure 2(b) illustrates the process of the estimation of the motion center.

Step 1 Compute sample mean μ and standard deviation σ based on all the samples of the distribution.

Step 2 Let $\mu_t(0) = \mu$ and $\delta = max(a\ \sigma, b * sampleSpaceWidth)$ where a and b are scaling factors, e.g., $a = 1.0$ and $b = 0.2$, and $sampleSpaceWidth$ is the image-width and image-height in the x and y direction, respectively.

Step 3 Compute trimmed mean $\mu_t(k+1)$ based on the samples within the interval $[\mu_t(k) - \delta, \mu_t(k) + \delta]$.

Step 4 Repeat Step 3 until $|\mu_t(k+1) - \mu_t(k)| < \epsilon$ where ϵ is the tolerance, e.g., $\epsilon = 1.0$. Denote the converged mean as μ^*.

Step 5 Let center-position $= \mu^*$.

In addition to the robust estimation of object center position, we propose the following routine for robust estimation of object size. The method first re-projects the frame difference result in a neighborhood of the located center. It then derives the object size based on the robust (trimmed) standard deviation. Given the robust mean μ^* and δ obtained from the above center locating routine, the routine for estimation the size in either x or y direction is as follows.

Step 1 Construct a clipped projection histogram H^{clip} by projecting the color filtering map within the range $[\mu^*_{opp} - \Delta, \mu^*_{opp} + \Delta]$ in the opposite direction, where μ^*_{opp} is the robust mean in the opposite direction and Δ determines the number of samples used in the calculation.

Step 2 Based on H^{clip}, compute the trimmed standard deviation δ_t based on the samples within the interval $[\mu^* - \delta, \mu^* + \delta]$.

Step 3 IF $H^{clip}(\mu^* + d\delta_t) \geq g\ H^{clip}(\mu^*)$ OR $H^{clip}(\mu^* - d\delta_t) \geq g\ H^{clip}(\mu^*)$,
where e.g., $d = 1.0$ and $g = 0.4$, THEN increase δ_t until the condition is no longer true.

Step 4 Let $size = c\ \delta_t$ where c is a scaling factor, e.g., $c = 2.0$.

B. Texture and Color Analysis: Low-Level Descriptors

To obtain rich, and hence robust and expressive descriptions of the objects in the video frames we describe each pixel in terms of color and texture measures. The color measures are the normalized red, green, and blue intensities of the pixel, and its grayvalue, while the texture measures are derived from the Gray Level Co-occurrence Matrix (GLCM), Fractal Dimension estimation methods, and a Gabor filter bank. The feature space representations of each pixel are classified into the categories sky/clouds, grass, trees, animal, rock using a back-propagation neural network. The use of these features in conjunction with the back-propagation classifier have previously been shown to enable the detection of deciduous trees in unconstrained images [14].

The rich image descriptions are formed from 56 Gray-Level Co-occurrence Matrix, 4 fractal dimension, 12 Gabor, and 4 color based measures. No one of the types of measure (e.g. color or Gabor measures) has the power of the combined set of measures. The neural network described in Section II-C is well suited to combine this set of measures and robustly classify image regions into various animal and non-animal classes. Note that we are only computing features from still frames and that motion is included explicitly at a higher level. In an alternative approach [27] uses temporal textures for classification, by combining spatial and temporal changes in image sequences.

B.1 Gabor Filter Measures

The image (in the spatial domain) is described by its 2-D intensity function. The Fourier Transform of an image represents the same image in terms of the coefficients of sine and cosine basis functions at a range of frequencies and orientations. Similarly, the image can be expressed in terms of coefficients of other basis functions. Gabor [13] used a combined representation of space and frequency to express signals in terms of "Gabor" functions:

$$F_{\theta,\nu}(\mathbf{x}) = \sum_{i=1}^{n} a_i(\mathbf{x})\, g_i(\theta, \nu) \tag{1}$$

where θ represents the orientation and ν the frequency of the complex Gabor function:

$$g_i(\theta, \nu) = e^{i\nu(xcos(\theta)+ysin(\theta))} e^{-\frac{x^2+y^2}{\sigma^2}} \tag{2}$$

Gabor filters have gained popularity in multi-resolution image analysis [12], [13], despite the fact that they do not form an orthogonal basis set. Gabor filter based wavelets have recently been shown [21] to be fast and useful for the retrieval of image data.

We convolve each image with Gabor filters tuned to four different orientations at 3 different scales. The average and range of the four measures at each scale are computed. To make the measurements somewhat scale invariant, we obtain the following four texture measures:

- The average of the orientation responses at all scales.
- The average of the scales' orientation response range.
- The range of the scales' averaged orientation responses.
- The range of the scales' orientation response range.

B.2 Graylevel Co-occurrence Matrix Measures

Let $p(i, j, d, \theta) = \frac{P(i,j,d,\theta)}{R(d,\theta)}$ where $P(.)$ is the graylevel co-occurrence matrix of pixels separated by distance d in orientation θ and where $R(.)$ is a normalization constant that causes the entries of $P(.)$ to sum to one.

In texture classification, the following measures have been defined, see for example [4], [15]:

The **Angular Second Moment (E)** (also called the Energy) assigns larger numbers to textures whose co-occurrence matrix is sparse.

$$E(d,\theta) = \sum_{j=1}^{N_g}\sum_{i=1}^{N_g}[p(i,j,d,\theta)]^2$$

The **Difference Angular Second Moment (DASM)** assigns larger numbers to textures containing only a few graylevel patches. This and other features use $p_{x-y}(n,d,\theta) = \sum_{\substack{j=1 \\ |i-j|=n}}^{N_g}\sum_{i=1}^{N_g} p(i,j,d,\theta)$

$$DASM(d,\theta) = \sum_{n=0}^{N_g} p_{x-y}(n,d,\theta)^2$$

The **Contrast (Con)** is the moment of inertia around the co-occurrence matrix's main diagonal. It is a measure of the spread of the matrix values and indicates whether pixels vary smoothly in their local neighborhood.

$$Con(d,\theta) = \sum_{n=0}^{N_g-1} n^2 \left[\sum_{\substack{j=1 \\ |i-j|=n}}^{N_g}\sum_{i=1}^{N_g} p(i,j,d,\theta) \right]$$

The **Inverse Difference Moment (IDM)** measures the local homogeneity of a texture. It weighs the contribution of the co-occurrence matrix entries inversely proportional to their distance to the main diagonal.

$$IDM(d,\theta) = \sum_{i=1}^{N_g-1}\sum_{j=1}^{N_g-1} \frac{1}{1-(i-j)^2} p(i,j,d,\theta)$$

The **Mean (M)** is similar to the contrast measure above but weights the off-diagonal terms linearly with the distance from the main diagonal, rather than quadratically as for the Contrast.

$$M(d,\theta) = \sum_{n=0}^{N_g-1} n \left[\sum_{\substack{j=1 \\ |i-j|=n}}^{N_g}\sum_{i=1}^{N_g} p(i,j,d,\theta) \right]$$

Similar to the Angular Second Moment the **Entropy (H)** is large for textures that give rise to co-occurrence matrices whose sparse entries have strong support in the image. It is minimal for matrices whose entries are all equally large.

$$H(d,\theta) = -\sum_{j=1}^{N_g}\sum_{i=1}^{N_g} p(i,j,d,\theta)\log\left(p(i,j,d,\theta)\right)$$

Other measures are, **Sum Entropy (SH)**, which uses $p_{x+y}(n,d,\theta) = \sum_{j=1}^{N_g}\sum_{\substack{i=1 \\ |i+j|=n}}^{N_g} p(i,j,d,\theta)$

$$SH(d,\theta) = -\sum_{n=0}^{2*N_g-1} p_{x+y}(n,d,\theta)\log\left(p_{x+y}(n,d,\theta)\right)$$

Difference Entropy (DH)

$$DH(d,\theta) = -\sum_{n=0}^{N_g} p_{x-y}(n,d,\theta)\log\left(p_{x-y}(n,d,\theta)\right)$$

Difference Variance (DV)

$$DV = -\sum_{n=2}^{2N_g}(n-DH)^2 p_{x-y}(n,d,\theta)$$

The **Correlation (Cor)** measure is an indication of the linearity of a texture. The degree to which rows and columns resemble each other strongly determines the value of this measure. This and the next two measures use $\mu_x = \sum_i i \sum_j p(i,j,d,\theta)$ and $\mu_y = \sum_j j \sum_i p(i,j,d,\theta)$.

$$Cor(d,\theta) = \frac{\sum_{i=1}^{N_g-1}\sum_{j=1}^{N_g-1} ijp(i,j,d,\theta) - \mu_x * \mu_y}{\sigma^2}$$

Shade (S)
$$S(d,\theta) = \sum_{i}^{N_g}\sum_{j}^{N_g}(i+j-\mu_x-\mu_y)^3 p(i,j,d,\theta)$$

Prominence (P)
$$P(d,\theta) = \sum_{i}^{N_g}\sum_{j}^{N_g}(i+j-\mu_x-\mu_y)^4 p(i,j,d,\theta)$$

Note that the directionality of a texture can be measured by comparing the values obtained for a number of the above measures as θ is changed. The above measures were computed at $\theta = \{0^o, 45^o, 90^o, \text{ and } 135^o\}$ using $d = 1$. For further discussion of these graylevel co-occurrence matrix measures, see [4], [15], [29].

B.3 Fractal Dimension Measures

The underlying assumption for the use of the Fractal Dimension (FD) for texture classification and segmentation is that images or parts of images are self-similar at some scale.

Various methods that estimate the FD of an image have been suggested:

- Fourier-transform based methods [23],
- box-counting methods [3], [19], and
- 2D generalizations of Mandelbrot's methods [22].

The principle of self-similarity may be stated as: If a bounded set A (object) is composed of N_r non-overlapping copies of a set similar to A, but scaled down by a reduction factor r, then A is self-similar. From this definition, the Fractal Dimension D is given by

$$D = \frac{log\, N_r}{log\, r}$$

The FD can be approximated by estimating N_r for various values of r and then determining the slope of the least-squares linear fit of $\frac{log\, N_r}{log\, \frac{1}{r}}$. The differential box-counting method outlined in Chaudhuri, *et al* [3] are used to achieve this task.

Three features are calculated based on

- the actual image patch $I(i,j)$,
- the high-graylevel transform of $I(i,j)$, $I_1(i,j) = \begin{cases} I(i,j) - L_1 & I(i,j) > L_1 \\ 0 & otherwise \end{cases}$
- the low-graylevel transform of $I(i,j)$, $I_2(i,j) = \begin{cases} 255 - L_2 & I(i,j) > 255 - L_2 \\ I(i,j) & otherwise \end{cases}$

where $L_1 = g_{min} + \frac{g_{avg}}{2}$, $L_2 = g_{max} - \frac{g_{avg}}{2}$, and g_{min}, g_{max}, and g_{avg} are the minimum, maximum and average grayvalues in the image patch, respectively.

The fourth feature is based on multi-fractals which are used for self-similar distributions exhibiting non-isotropic and inhomogeneous scaling properties. Let k and l be the minimum and maximum graylevel in an image patch centered at position (i,j), let $n_r(i,j) = l - k + 1$, and let $\mathcal{N}_r = \frac{n_r}{N_r}$, then the multi-fractal, D_2 is defined by

$$D_2 = \lim_{r \to 0} \frac{log\, \sum_{i,j} \mathcal{N}_r^2}{log\, r}$$

A number of different values for r are used and the linear regression of $\frac{log\, \sum_{i,j} \mathcal{N}_r^2}{log\, r}$ yields an estimate of D_2.

B.4 The Color Features

The final set of features are the 3 normalized color measures r, g, b and the intensity I

$$r = \frac{R}{R+G+B} \ , \quad g = \frac{G}{R+G+B} \ , \quad b = \frac{B}{R+G+B} \ , \quad I = \frac{R+G+B}{R_{max}+G_{max}+B_{max}}$$

We generally observed that although our feature set is theoretically redundant it is beneficial for the classifier to use all the measures rather than a carefully selected subset.

C. Region Classification and Motion Blob Verification

We use a back-propagation neural network to arbitrate between the different features describing the image. Our back-propagation neural network [10] has a single hidden layer and uses the sigmoidal activation function $\Phi(act) = \frac{1}{1+e^{-act}} - 0.5$, where act is the activation of the unit before the activation function is applied. A single hidden layer in a back-propagation neural network has been shown to be sufficient to uniformly approximate any function (mapping) to arbitrary precision [6]. Although this existential proof doesn't state that the best network for some task has a single hidden layer, we found one hidden layer adequate. The architecture of the network is shown in Figure 3. The back-propagation algorithm propagates the (input) function values layer by layer, left to right (input to output) and back-propagates the errors layer by layer, right to left (output to input). As the errors are propagated back to the input units, part of each unit's error is being corrected.

A number of factors prevent zero error results. A few of these complicating factors are that *often there is no correct classification.* For instance, should bushes be labeled as tree or non-tree areas? What if a bush is actually a small tree? In general it is difficult to label class border pixels correctly; and misclassifications need not all be equally important. Misclassifying a distant herd of animals as trees or rocks is not as severe a mistake as, for example, classifying a nearby lion as sky.

We trained the network using a total of 14 labels. 9 animal labels (lion, cheetah, leopard, antelope, impala, zebra, gnu, elephant, and an all-other-animal class) and 5 non-animal labels (rock, sky/clouds, grass, trees, and an all-other-non-animal class) as well as a don't care label that was used to tell the network to ignore border regions between instances of the different groups, which arguably are bad training inputs.

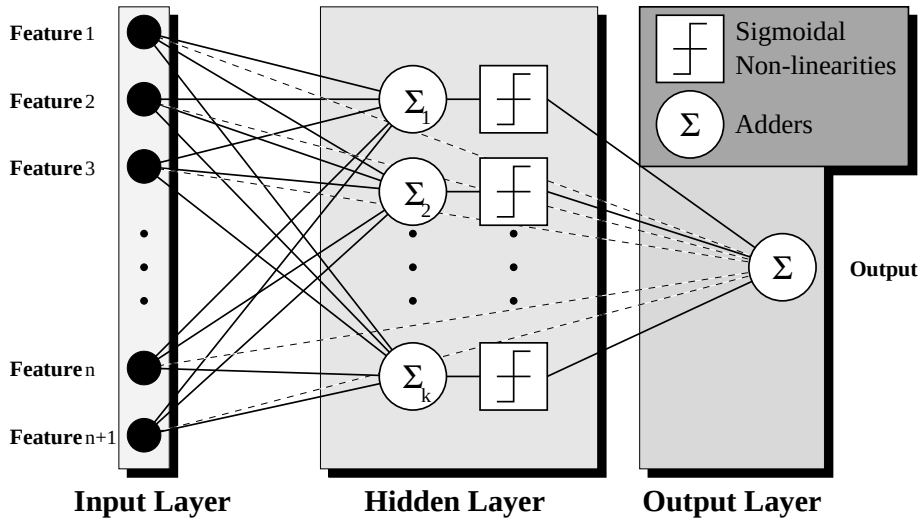


Fig. 3. The Neural Network architecture.

After training, we found that the proposed network performed well at classifying grass, trees, rocks, sky, and animals as a whole group. However, it is difficult for the network to classify lions, cheetahs, leopards, antelopes, impalas, gnus, hyenas, and even zebras, rhinos and elephants each into different groups. This is probably due to the fact that those animals differ mostly in their shape and size which we do not model. Hence, while the network was still trained on the different animal labels, we artificially grouped those labels into a single "animal" label when using the network for animal region verification. We also found that the network did not perform well at solving the opposite problem of classifying, grass, trees, rocks, and sky together as a single "non-animal" group. The differences between the appearance of instances of these groups are severe. Asking the network to assign one label to them and a different label to animals proves to be more difficult than the classification into the individual non-animal groups.

The output of the network is then used to verify the motion blob candidates from section II-A. In our current implementation, a simple procedure is employed which implements the following test. A region that has high residual motion after motion compensation and that contains a significant amount of animal labels, as detected by the neural network, is considered as a possible moving animal region.

D. Shot Summarization and Intermediate-Level Descriptors

We use a simple color histogram based technique to decompose video sequences into shots. To avoid missing important events in extended shots, we also *force* a shot summary every 200 frames. A third kind of shot boundary is inserted whenever the direction of the global motion changes. Shot boundaries of this last kind ensure that the motion within shots is homogeneous. Each shot is then summarized in terms of intermediate-level descriptors. The purpose of generating intermediate-level shot summaries is two-fold. First, the shot summaries provide a way to encapsulate the low-level feature and motion analysis details so that the high-level event inference module may be developed independent of those details, rendering it robust against implementational changes. Second, the shot summaries abstract the low-level analysis results so that they can be read and interpreted more easily by humans. This simplifies the algorithm development process and may also facilitate video indexing, retrieval and browsing in video database applications.

In general, the intermediate-level descriptors may consist of (1) *object*, (2) *spatial*, and (3) *temporal* descriptors. The *object* descriptors, e.g., "animal", "tree", "sky/cloud", "grass", "rock", etc. indicate the existence of certain objects in the video frames. The *spatial* descriptors represent the location and size information about objects and the spatial relations between them in terms of spatial prepositions such as "inside", "next to", "on top of", etc. [7], [8]. The *temporal* descriptors represent motion information about objects and the temporal relations between them in terms of temporal prepositions such as "while", "before", "after", etc. [7], [8].

For the hunt detection application, we currently employ a particular set of intermediate-level descriptors which describe: (1) whether the shot summary is due to a forced or detected shot boundary; (2) the frame number of the beginning of the shot; (3) the frame number of the end of the shot; (4) the global motion; (5) the object motion; (6) the initial object location; (7) the final object location; (8) the initial object size; (9) the final object size; (10) the smoothness of the motion; (11) the precision throughout shot; and (12) the recall throughout shot. More precisely, the motion descriptors provide information about the x- and y- translation and zoom components of motion. The location and size descriptors indicate the location and size of the detected dominant motion blob at the

beginning and the end of the shot. The precision is the average ratio of the number of animal labels within the detected dominant motion blob versus the size of the blob, while the recall is an average of the ratio of the animal labels within the detected dominant motion blob versus the number of animal labels in the entire frame. In addition, we also employ descriptors indicating (13) that tracking is engaged; (14) that object motion is fast; (15) that an animal is present; (16) the beginning of a hunt; (17) number of consecutive hunt shot candidates found; (16) the end of a hunt; and (19) whether a valid hunt is found. See Section III-F for an example and further explanation.

E. Event Inference

Hunt events are detected by an event inference module which utilizes domain-specific knowledge and operates at the shot level based on the generated shot summaries. From observation and experimentation with a number of wildlife documentaries, a set of rules have been deduced for detecting hunts. The rules reflect the fact that a hunt usually consists of a number of shots exhibiting smooth but fast animal motion which are followed by subsequent shots with slower or no animal motion. In other words, the event inference module looks for a prescribed number of shots in which (a) there is at least one animal of interest; (b) the animal is moving in a consistently fast manner for an extended period; and (c) the animal stops or slows down drastically after the fast motion. Figure 4 shows and describes a state diagram of our hunt detection inference model.

Automatic detection of the properties and sequences of actions in the state digram is non-trivial and the low-level feature and motion analysis described earlier in this paper are necessary to realize the inference. Since any event can be defined by the occurrence of objects involved and the specification of their spatio-temporal relationship, the proposed mechanism, of combining low-level visual analysis and high-level domain-specific rules, may be applicable to detect other events in different domains. In Section III-G, we provide an example and further explanation for using this inference model for hunt detection.

III. Experimental Results

The proposed algorithm has been implemented in C++ and tested on Sun workstations. To evaluate the effectiveness of the algorithm, we have digitized wildlife video footage from

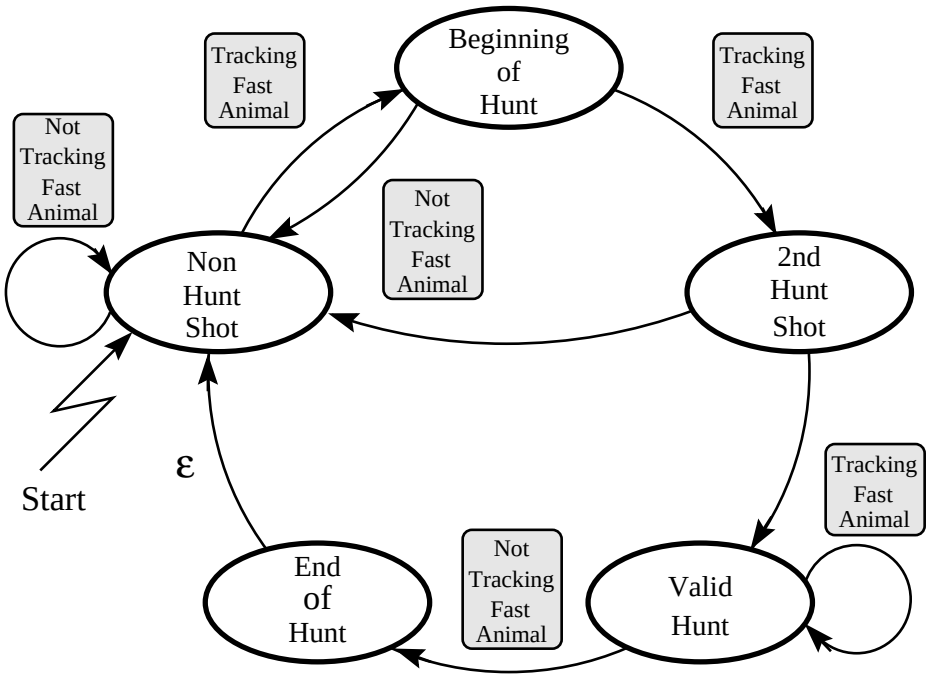


Fig. 4. The state diagram of our hunt detection method. Initially the control is in the Non-Hunt state on the left. When a fast moving animal is detected the control moves to the Beginning of Hunt state at the top of the diagram. When three consecutive shots are found to track fast moving animals then the Valid Hunt flag is set. The first shot afterwards that does not track a fast moving animal takes the control to the End of Hunt state, before again returning to the Non-Hunt state.

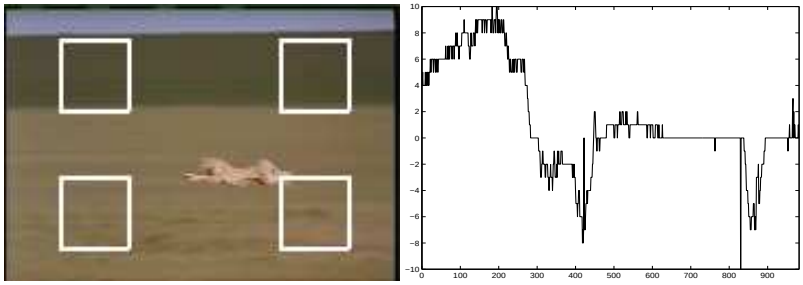
a number of commercially available VHS tapes from different content providers. In the following sections we show examples of the extracted texture and color features, the motion estimation and detection results, the region classification results, the shot summaries, and the final hunt event detection results.

A. Test Data

About 45 minutes of actual wildlife video footage have been digitized and stored as test data for our hunt detection experiments. The frame rate of the video is 30 frames per second and the digitized frame resolution is 360 x 243 pixels. A total of 10 minutes of footage $\stackrel{\triangle}{=}$ 18000 frames $\stackrel{\triangle}{=}$ 100 shots have been processed so far.

B. Global Motion Estimation

Figure 5(a) shows the size and locations of the four regions at which the global motion is estimated. For each pair of frames motion estimates are computed using a 5 level pyramid scheme at the shown patch locations. In addition the previous motion estimate is taken as the current motion estimate and a tight local search around the four *predicted* patch locations yields another four patch matches. The best match of any of these 8 patch comparisons becomes the motion estimate for the current frame pair. Figure 5(b) shows the motion estimates during a hunt.



(a) (b)

Fig. 5. (a) The locations used to estimate the global motion, and (b) the motion estimates during a hunt.

C. Motion Blob Detection

Figure 6 shows an example of the motion blob detection results. It is apparent that reliable estimation and compensation of global motion makes the task of motion blob detection relatively easier. When the accuracy of the global motion estimation results are poor, the performance of the motion blob detection relies largely on the robustness of the motion blob detection and tracking algorithm described in Section 2.1.

D. Feature Space Representation of the Video Frames

Figure 7 shows the feature space representation of a video frame. The features shown are the results of the Gray-Level Co-occurrence Matrix based measures (first 56 feature images), the Fractal Dimension based measures (next 4 feature images), the color based measures (next 4 feature images), and the Gabor based measures (last 12 feature images).

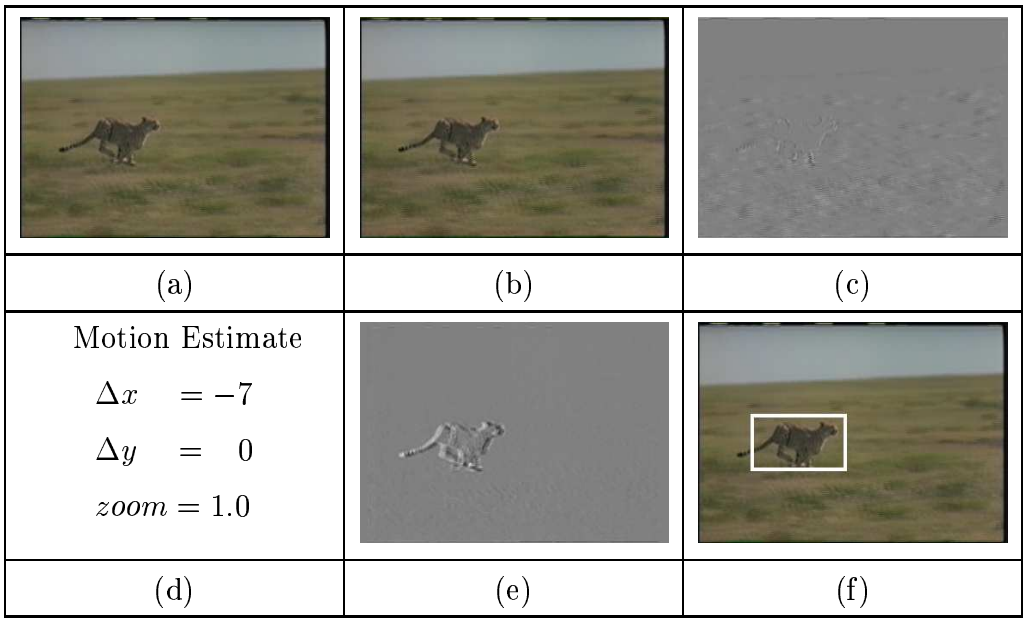


Fig. 6. Two consecutive frames from a hunt (a) and (b), the difference image (c), the estimated motion between the two frames (d), the motion compensated difference image (e), and the box around the area of largest residual error in the motion compensated difference image.

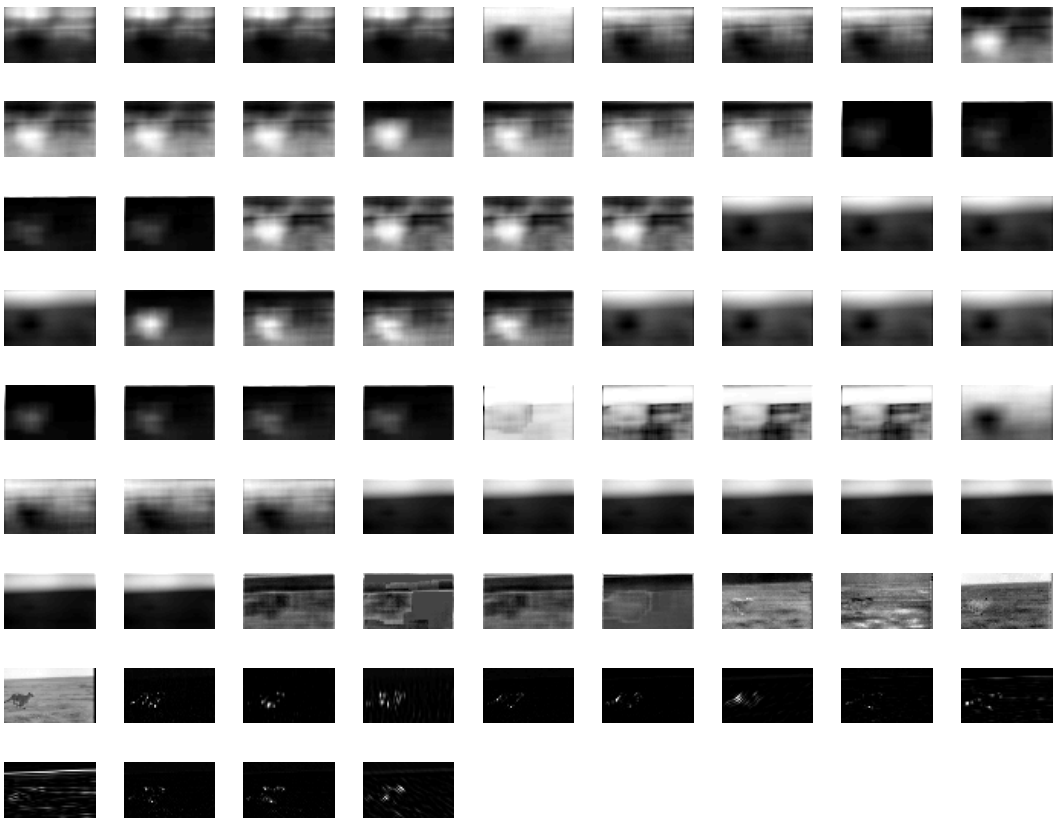


Fig. 7. The feature space representation of the first frame in Figure 6.

E. Region Classification

A neural network is trained on a number of training frames from wildlife video. The network is then used to classify unseen wildlife video. Global motion estimates such as the ones in Figure 5 are used to detect moving objects as shown in Figure 6. The locations of these moving object blobs are then verified using a neural network image region classifier that combines color and texture information. Rows 1, 3, and 5 of Figure 8 show a number of frames from hunts together with their classification results (rows 2, 4, and 6).

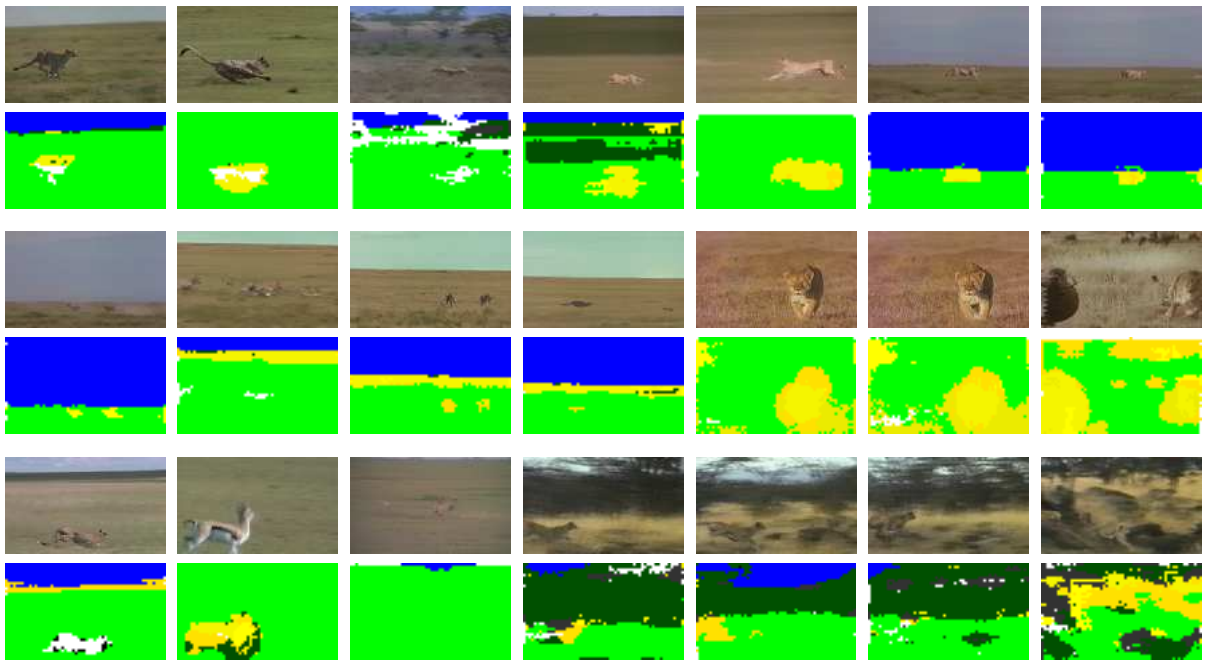


Fig. 8. Color and texture based segmentation results.

F. Shot Summarization

The intermediate level process consists of two stages. In the first stage the global motion estimates are analyzed and directional changes are detected in the x and y directions. When the *signs* of the 50 frame global motion averages before and after the current frame differ and their *magnitudes* are greater than 1 pixel per frame we insert an artificial shot boundary. In the second stage each shot is then summarized as in the example shown below.

---------- General Information ----------		------- Hunt Information -------	
Forced/real shot summary	: 0	Tracking	: 1
First frame of shot	: 64	Fast	: 1
Last frame of shot	: 263	Animal	: 1
Global motion estimate (x,y)	: (-4.48, 0.01)	Beginning of hunt	: 1
Within frame animal motion estimate (x,y)	: (-0.17, 0.23)	Number of hunt shot candidates	: 1
Initial position (x,y)	: (175,157)	End of hunt	: 0
Final position (x,y)	: (147,176)	Valid hunt	: 0
Initial size (w,h)	: (92, 67)		
Final size (w,h)	: (100, 67)		
Motion smoothness throughout shot (x,y)	: (0.83, 0.75)		
Precision throughout shot	: (0.84)		
Recall throughout shot	: (0.16)		

The summary consists of two parts, the first part, under `General Information` shows general statistics extracted for this shot, while the second, under `Hunt Information` consists of inferences based on those statistics for the hunt detection application.

The first row of the general Information part of the summary shows whether the shot boundary corresponding to this shot summary was real, i.e. whether it was detected by the shot boundary detector, or if it was forced because the maximum number of frames per shot was reached or the global motion has changed. The next two rows show the first and last frame numbers of this shot. The following measurements are shot statistics, i.e., the average global motion over the entire shot on row four, and the average object motion within the shot on row five. The next four rows measure the initial position and size, as well as the final position and size of the detected dominant motion blob. The third last row shows the smoothness of global motion where values near 1 indicate smooth motion and values near 0 indicate unstable motion estimation. The detection of a reversal of the global motion direction, described above, was based on a long term average of the motion estimates around the current frame, indicates a *qualitative* change in the global motion. The smoothness measure described here, on the other hand, provides a *quantitative* measure of the smoothness of the estimated motion. Finally the last two rows show the average precision and recall for the entire shot. As defined in Section II-D, the precision is the average ratio of the number of animal labels within the detected dominant motion blob versus the size of the blob, while the recall is an average of the ratio of the animal labels within the detected dominant motion blob versus the number of animal

labels in the entire frame.

The hunt information part of the shot summary shows a number of predicates that were inferred from the statistics in part one. The shot summary shown above summarizes the first hunt shot following a `forced` shot boundary. The system is indicating that it is `Tracking` a `Fast` moving `Animal` and hence, that this could be the `Beginning of a hunt`. The `Tracking` predicate is true when the motion smoothness measure is greater than a prescribed value and the motion blob detection algorithm detects a dominant motion blob. The `Fast` predicate is set to true if the translational components of the estimated global motion are sufficiently large in magnitude, and the `Animal` predicate is true if the precision, i.e. the number of animal labels within the tracked region, is sufficiently large. (The recall measure has not been used in our current implementation.) The remaining predicates are determined and used by the inference module as described below.

G. Event Inference and Final Detection Results

The event inference module infers the occurrence of a hunt based on the intermediate descriptors as described in Section III-F. In doing so, it employs four predicates, `Beginning of hunt`, `Number of hunt shot candidates`, `End of hunt`, and `Valid hunt`, which are currently embedded in the shot summary. If the intermediate descriptors `Tracking`, `Fast` and `Animal` are all true for a given shot, the inference module sets `Beginning of hunt` to be true, which means the shot could potentially be the beginning of a hunt event. The inference module tracks the intermediate descriptors `Tracking`, `Fast` and `Animal` for consecutive shots and increments the value of the `Number of hunt shot candidates` if all those three descriptors hold true for consecutive shots. In our current implementation, when the `Number of hunt shot candidates` is equal or greater than 3, `Valid hunt` is set to be true. Finally the inference module sets `End of hunt` to be true if one of the intermediate descriptors `Tracking`, `Fast` and `Animal` becomes false, which implies either the animal is no longer visible or trackable, or the global motion is slow enough indicating a sudden stop after fast chasing.

In our final results, hunt events are specified in terms of their starting and ending frame numbers. In the 10 minutes (18000 frames) of wildlife video footage which we have processed, there exist 7 hunt events. Table I shows the actual frames of the 7 hunts and

all the frames of the detected hunts when we applied the proposed algorithm to the 10 minute video footage. The table also shows the retrieval performance of our method in terms of the two commonly used evaluation criteria (1) precision and (2) recall.

TABLE I

A COMPARISON OF THE ACTUAL AND DETECTED HUNTS IN TERMS OF THE FIRST AND LAST HUNT FRAME, AND THE ASSOCIATED PRECISION AND RECALL.

Sequence Name	Actual Hunt Frames	Detected Hunt Frames	Precision	Recall
hunt1	305 - 1375	305 - 1375	100 %	100 %
hunt2	2472 - 2696	2472 - 2695	100 %	99.6%
hunt3	3178 - 3893	3178 - 3856	100 %	94.8%
hunt4	6363 - 7106	6363 - 7082	100 %	96.8%
hunt5	9694 - 10303	9694 - 10302	100 %	99.8%
hunt6	12763 - 14178	12463 - 13389	67.7%	44.2%
hunt7	16581 - 17293	16816 - 17298	99.0%	67.0%
Average			95.3%	86.0%

IV. Summary and Discussion

In this paper, we have presented a new computational framework and a number of enabling algorithmic components for automatic event detection in video and applied it to detect hunts in wildlife documentaries. Our experimental results have verified the effectiveness of the proposed algorithm. The developed framework decomposes the task of extracting semantic events into three stages where visual information is analyzed and abstracted. The first stage extracts low-level features and is entirely domain-independent. The second stage analyzes the extracted low-level features and generates intermediate-level descriptors some of which may be domain-specific. In this stage, shots are summarized in terms of both domain-independent and domain-specific descriptors. To generate the shot summaries, regions of interest are detected, verified and tracked. The third and final stage is domain-specific. Rules are deduced from specific domains and an inference model is built based on the established rules. In other words, each lower stage encapsulates certain low-level visual processing from the higher stages. Therefore the processes in the higher stages

can be stable and relatively independent of any potential detail changes in the lower level modules. In order to detect different events, the expected changes are (a) the addition of descriptors in the second stage and (b) the design of a new set of rules in the third stage. The proposed algorithm also provides several reusable algorithmic components. In fact, the extracted low-level texture and color features are domain independent and many objects involved in events carry certain texture and color signatures. The neural network used for image region classification can be easily re-configured or extended to handle other types of objects [14]. The robust statistical estimation based object tracking method has already been used in different applications and its robustness and simplicity are verified in experiments repeatedly [24].

It is important for us to point out that the proposed algorithm detects hunt events by detecting certain spatial-temporal phenomena which are physically associated with a hunt event in the nature. More precisely, the physical phenomenon which we attempt to capture is the combination of the presence of animals in space and their movement patterns in time. This is in contrast to many existing event detection methods which detect certain events by detecting artificial postproduction editing patterns or other artifacts. The drawbacks of detecting specific editing patterns or other artifacts are that those patterns are often content provider dependent and it is difficult, if not impossible, to modify the detection methods and apply them to the detection of other events. It is also important to point out that our algorithm solves a practical problem and the solution is needed in the real world. In the wildlife video tapes which we obtained, the speech from the audio track and the text from the close-caption are loosely correlated with the visual footage. It is therefore unlikely that the hunt segments may be accurately located by analyzing the audio track and close-caption. In other words, given the existing wildlife tapes, a visual-information-based detection algorithm is needed to locate the hunt segments otherwise manual annotation is required.

An immediate focus of future work is to develop a full set of intermediate-level descriptors for generating shot summaries. The purpose of developing the descriptors is to provide a wider coverage over different domains and events so that fewer domain-specific descriptors need to be added in new applications. Other future work is to improve the

procedure which detects and tracks regions of interest. Finally, it would be interesting to adopt machine learning techniques into the event inference engine so that it can improve its performance over time automatically.

References

[1] F. Arman, R. Depommier, A. Hsu, and M.-Y. Chiu, "Content-based Browsing of Video Sequences," *Proc. ACM Multimedia*, pp. 97-103, 1994.

[2] S.-F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong, "A Fully Automated Content Based Video Search Engine Supporting Spatio-Temporal Queries," *IEEE Trans. Circuits and Systems for Video Technology*, 1998.

[3] B.B. Chaudhuri, N. Sarkar, and P. Kundu, "Improved Fractal Geometry Based Texture Segmentation Technique," *IEE Proceedings*, part E, vol. 140, pp. 233-241, 1993.

[4] R.W. Conners, C.A. Harlow, "A Theoretical Comparison of Texture Algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 2, no 3, pp. 204-222, 1980.

[5] J. D. Courtney, "Automatic Video Indexing via Object Motion Analysis," *Pattern Recognition*, vol. 30, no. 4, pp. 607-626, 1997.

[6] G. Cybenko, "Approximation by Superposition of Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, Chapter 2, pp. 303-314, 1989.

[7] A. Del Bimbo, E. Vicario, D. Zingoni, "A Spatial Logic for Symbolic Description of Image Contents," *J. Visual Languages and Computing*, vol. 5, pp. 267-286, 1994.

[8] N. Dimitrova and F. Golshani, "Motion Recovery for Video Content Classification," *ACM Trans. Information Systems*, vol. 13, no 4, pp 408-439, 1995.

[9] P. England, R.B. Allen, M. Sullivan, and A. Heybey, "I/Browse: The Bellcore Video Library Toolkit," *SPIE Proc. Storage and Retrieval for Image and Video Databases*, pp. 254-264, 1996.

[10] S. Fahlman, "Faster-Learning Variations on Back-Propagation: An Empirical Study," *Proc. Connectionist Models Summer School*, Morgan Kaufmann, 1988.

[11] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*, MIT Press, 1993.

[12] I.Fogel and D.Sagi, "Gabor Filters as Texture Discriminator," *J. Biological Cybernetics*, vol. 61, pp. 103-113, 1989.

[13] D. Gabor, "Theory of communication," *J. IEE*, vol. 93, pp. 429-457, 1946.

[14] N. Haering, Z. Myles, and N. da Vitoria Lobo, "Locating Deciduous Trees," *Proc. IEEE Workshop on Content-based Access of Image and Video Libraries*, pp. 18-25, 1997.

[15] R.M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification," *IEEE Trans. Systems Man and Cybernetics*, vol. 3, no 6, pp. 610-621, 1973.

[16] S. S. Intille, "Tracking Using a Local Closed-World Assumption: Tracking in the Football Domain," *Master Thesis, M.I.T. Media Lab*, 1994.

[17] G. Iyengar and A. Lippman, "Models for Automatic Classification of Video Sequences", *SPIE Proc. Storage and Retrieval for Image and Video Databases*, pp. 216-227, 1997.

[18] T. Kawashima, K. Tateyama, T. Iijima, and Y. Aoki, "Indexing of Baseball Telcast for Content-based Video Retrieval," *Proc. International Conference on Image Processing*, pp. 871-875, 1998.

[19] J.M. Keller and S. Chen, "Texture Description and Segmentation through Fractal Geometry," *Computer Vision, Graphics and Image Processing*, vol. 45, pp. 150-166, 1989.

[20] R. L. Lagendijk, A. Hanjalic, M. Ceccarelli, M. Soletic, and E. Persoon, "Visual Search in a SMASH System", *Proc. International Conference on Image Processing*, pp. 671-674, 1997.

[21] B.Manjunath and W.Ma, "Texture Features for Browsing and Retrieval of Image Data," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837-859, 1996.

[22] S. Peleg, J. Naor, R. Hartley, and D. Avnir, "Multiple Resolution Texture Analysis and Classification," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no 4, pp. 518-523, 1984.

[23] A.P. Pentland, "Fractal-based Description of Natural Scenes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no 6, pp. 661-674, 1984.

[24] R. J. Qian, M. I. Sezan and K. E. Matthews, "A Robust Real-Time Face Tracking Algorithm", *Proc. International Conference on Image Processing*, pp. 131-135, 1998.

[25] D. Saur, Y.-P. Tan, S.R. Kularni, and P.J. Ramadge, "Automated Analysis and Annotation of Basketball Video," *SPIE Proc. Storage and Retrieval for Image and Video Databases*, pp. 176-187, 1997.

[26] M. Smith and T. Kanade, "Video Skimming for Quick Browsing Based on Audio and Image Characterization," *CMU Computer Science Department Technical Report CMU CS-95-186*, 1995.

[27] M. Szummer, "Temporal Texture Modeling," *Master Thesis, M.I.T. Media Lab*, 1995.

[28] N. Vasconcelos and A. Lippman, "A Bayesian Framework for Semantic Content Characterization," *Proc. Computer Vision and Pattern Recognition*, pp. 566-571, 1998.

[29] J.S. Weszka, C.R. Dyer, and A. Rosenfeld, "A Comparative Study of Texture measures for Terrain Classification," *IEEE Trans. Systems Man and Cybernetics*, vol. 6, no 4, pp. 269-285, 1976.

[30] R.R. Wilcox, *Introduction to Robust Estimation and Hypothesis Testing*, Statistical Modeling and Decision Science Series, Academic Press, 1997.

[31] M. Yeung, and B.-L. Yeo, "Video Visualization for Compact Presentation and Fast Browsing of Pictorial Content," *IEEE Trans. Cicuits and Systems for Video Technology*, vol. 7, no 5, pp. 771-785, 1996.

[32] D. Yow, B.L.Yeo, M. Yeung, and G. Liu, "Analysis and Presentation of Soccer Highlights from Digital Video," *Proc. Asian Conference on Computer Vision*, 1995.

[33] H. J. Zhang, S. W. Smoliar, and J. H. Wu, "Content-Based Video Browsing Tools," *SPIE Proc. Storage and Retrieval for Image and Video Databases*, pp. 389-398, 1995.